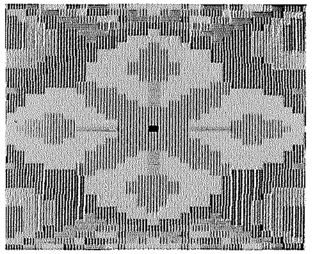
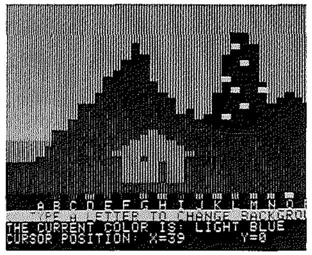
apple computer inc.®

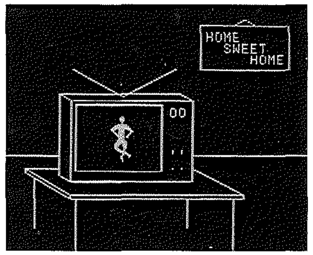
10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010

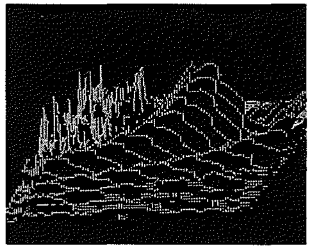
THE APPLE TAPES
INTRODUCTORY PROGRAMS FOR THE APPLE II

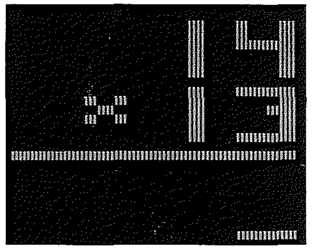


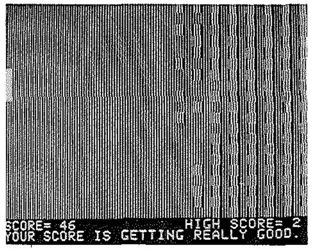


```
   A B C D E F G H I J K L M N O P
   TYPE A LETTER TO CHANGE BACKGROU
THE CURRENT COLOR IS: LIGHT BLUE
CURSOR POSITION: X=39         Y=0
```

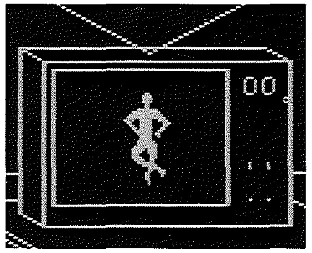


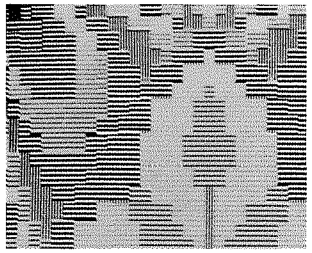


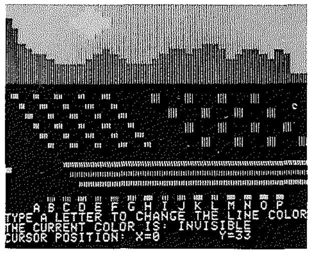




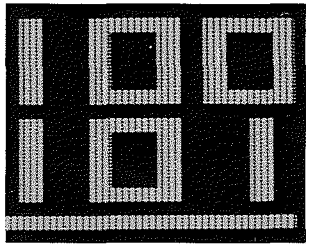
```
SCORE= 46               HIGH SCORE= 2
YOUR SCORE IS GETTING REALLY GOOD.
```

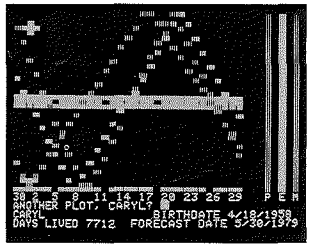




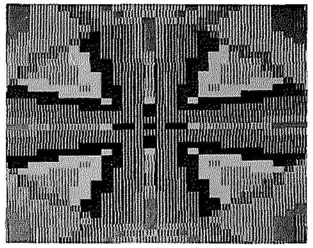


```
    A B C D E F G H I J K L M N O P
TYPE A LETTER TO CHANGE THE LINE COLOR
THE CURRENT COLOR IS: INVISIBLE
CURSOR POSITION: X=0         Y=33
```





```
30 2  5  8  11 14 17 20 23 26 29   P E M
ANOTHER PLOT, CARYL?
CARYL                  BIRTHDATE 4/18/1958
DAYS LIVED 7712  FORECAST DATE 5/30/1979
```



apple computer inc.®

THE APPLE TAPES
INTRODUCTORY PROGRAMS FOR THE APPLE II

PREFACE

This booklet contains documentation for the programs that came with your Apple II system. These programs illustrate some of the capabilities available to you with your new Apple. You will understand these programs better if you read the Apple BASIC Programming Manual first, but that is not necessary to your enjoyment of the programs. To use any Apple program on cassette tape, LOAD the tape in the following manner.

The Usual Procedure for Loading Tapes

1. Make sure your computer is in Integer BASIC
2. Rewind the tape
3. Start the tape playing
4. Type LOAD
   After you press RETURN the cursor will disappear. Nothing happens for from five to twenty seconds, and then the Apple beeps. This means that the tape's information has started to go into the computer. After some more time (depending on how much information is on the tape, but usually less than a few minutes) the Apple beeps again and the prompt character and the cursor reappear.

4a. If you got an error message such as ERR , the tape did not LOAD properly. Begin the tape LOADing process again at step one.

5. Stop the tape recorder and rewind the tape. The information has been transferred, and you are finished with the tape recorder for the time being.

NOTE: If you have some experience as a programmer, it may interest you to know that Programmer's Aid #1 is installed on the main board of your new Apple. Refer to the Programmer's Aid #1 manual for information on its uses.

Enjoy your new Apple!

TABLE OF CONTENTS

ALIGNMENT TEST TONE

The Alignment Test Tone tape does not contain a program. It contains a high-frequency tone with which you can align your tape recording head. If you have problems loading programs into your Apple from cassette tape, your recording head may need aligning.

It is possible that your problem loading programs from cassette tape is being caused by a mechanical malfunction. Before using this tape, check that your tape recorder is mechanically sound and your Apple is working properly. You should be aware that tapes recorded with a poorly aligned head will not work properly when played back on a recorder with a properly aligned one. Before you use the Alignment Test Tone tape, it is a good idea to LOAD any programs you may have saved with your poorly aligned recorder and then reSAVE them with a recorder that you know is working well.

You will need the following articles in order to align your tape recording head with this method:

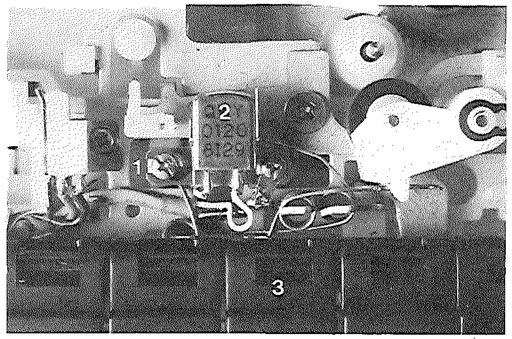
1. The Alignment Test Tone cassette tape.
2. A non-conductive surface to work on, such as a wooden table or desk. If you work on metal or any other conductive surface, you may get an electrical shock, or damage your recorder's electronics.
3. A Phillips screw driver.
4. Batteries for your cassette recorder.

The Alignment Test Tone is not a program. In fact, your Apple isn't even used in the head-aligning process. Disengage your cassette recorder from your Apple, and rewind the Alignment Test Tone tape to the beginning. Now, unplug your recorder, and you are ready to begin.

Note: Your recording head can only be adjusted to <u>approximately</u> its ideal alignment with the Alignment Test Tone. For a more accurate adjustment, your tape recorder must be serviced by a trained technician.

The head-aligning process is really quite simple. The tape is played on the cassette recorder you wish to adjust. The high-pitched sound emitted by the tape is about 5 <u>kilohertz</u> or five thousand cycles per second. While the tape is playing, the <u>azimuth adjustment screw</u> is adjusted until the sound made by the tape is at its loudest. When the sound is as loud as it will go, the recording head is adjusted properly.

Cassette tape recorders are all a little different. On some of them the azimuth adjustment screw is easy to get to, while on others (such as the Panasonic) the outside cover must be removed from the recorder in order to locate the screw. Unplug your recorder, remove the Alignment Test Tone tape, and press the PLAY button. The recording head, (the shiny metal thing, usually centered just behind the control buttons) and some other metal and plastic parts, will move forward (toward the speaker). If you didn't see this in action, press the STOP button and then press the PLAY button again, watching carefully.



1 Azimuth Adjustment Screw

2 Recording Head

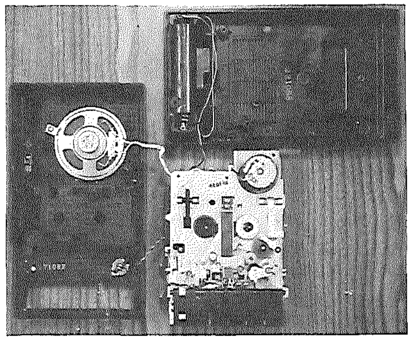
3 Control Buttons

Locate the recording head in the photograph, then locate the recording head on your own tape recorder. The rounded, highly polished surface on the recording head puts recorded information on the tape. This surface should not be touched. In fact, you should avoid touching all the parts on the inside of your recorder as much as possible. Oil from your skin can ruin some of these parts, and a slip of your finger could cause delicate parts to slip out of adjustment.

On most cassette recorders, you will have to remove the plastic casing that houses the recorder in order to get to the azimuth adjusment screw. Locate the azimuth adjustment screw in the photograph. The dark area surrounding the screw is a sealant to help keep the screw from slipping out of adjustment. The sealant is probably colored red on your recorder. If you can reach the azimuth screw well enough to adjust it with a screwdriver without removing the cover, you can skip the next two paragraphs, as they deal with removing the recorder's outside casing.

To remove the casing, turn the recorder bottom-up and remove all the screws on the bottom, including the one in the battery case underneath the batteries. (The Panasonic has five screws holding the case to the inner mechanism, counting the one in the battery case.) Make sure you know which screw fits in which hole so that they won't get mixed up when you put your recorder back together. Then gently remove the bottom casing, being careful of the wires that hold it to the inner mechanism. These wires are not very strong and some care must be taken to see that they are not damaged.

Now that the back has been removed, the handle and the top part of the casing should come right off. If they don't come off easily, look for screws on the front. Be careful not to remove any screws unless they attach the inner mechanism to the outside cover. When all the screws are out, the top casing can easily be removed. Again, be very careful of the wires. They are easily damaged. Arrange the parts of the recorder so that the inside section is laying flat on your work surface, and there is no tension on any of the wires.



Again find the azimuth adjustment screw in the first photograph. Then find the azimuth adjustment screw on your own tape recorder. Put batteries in your recorder's battery case, (wires connect the batteries to the recorder's inner mechanism), and play the Adjustment Test Tone tape. When you hear the high-pitched tone, adjust the azimuth adjustment screw with a phillips screw driver until the tone is as loud and piercing as you can make it. Put a little nail polish or white glue on the azimuth adjustment screw to help keep it from slipping out of adjustment. Don't move the recorder until the polish or glue is thoroughly dry (nail polish dries faster).

That's all there is to it. If you think you may have gotten skin oil on any sensitive part of the recorder, you can clean it carefully with rubbing alchohol and a cotton swab. Remove the tape and put your tape recorder back together, making sure to include the handle, and being very careful of the wires, of course.

SAMPLER

Submitted by: Bruce Tognazzini
Program Language: Integer BASIC

Ah, San Francisco, the romantic city. The lights, the cable cars, the Golden Gate Bridge. Many people journey far just to see this beatiful city. Whether you've been there or not, you ought to see Bruce Tognazzini's lovely needlepoint rendition of San Francisco.

"Needlepoint on a computer?" you say? Well, yes. And the needlepoint you can create on the Apple has dimensions your grandmother's needlework certainly never did. You'll notice that, in Bruce's needlepoint, lights flicker and the water of the Pacific ocean moves gently. You can almost hear the calm ocean lapping against the quiet shore. Try and top that with needle and thread!

To see Apple's needlepoint sampler, LOAD the tape marked SAMPLER and then type RUN . After the program title, copyright notice, and some brief instructions appear on the screen; just press the space bar and you will be rewarded with Bruce's vision of "The City".

BRICK OUT

Submitted by: Bruce Tognazzini
Program Language: Integer BASIC

Is your eye sharp? Is your hand quick? Brick Out will test your reflexes. To play Brick Out, first use the usual procedure to LOAD the tape marked BRICK OUT. Once the tape is LOADed, type RUN and the game title will appear on the screen followed by the all-important question "HI, WHAT'S YOUR NAME?". The name you type must be less than eighteen characters long (including spaces) because this program can't read words any longer than that. If you do type a name that is too long, and get an error message from your Apple, just type RUN and the program will begin over again.

After asking your name, the program will ask whether or not you prefer the standard colors. If you reply with a YES, the game board will appear on the screen in the standard colors. If you reply with a NO, all kinds of interesting things will happen. We won't tell you what kinds of things; you'll just have to try it and find out.

No matter what answer you give, be ready with one of the game controls. Your Apple will tell you if you are using the wrong one. Have fun!

COLOR DEMO

Submitted by: Jef Raskin and Rod Holt
Program Language: Integer BASIC

Color Demo is a low-resolution color graphics demonstration program. To see the Color Demo program do its stuff, LOAD the tape labeled COLOR DEMO, and then type RUN . A list of options numbered one through four will appear on your Apple's screen.

This kind of list is called a "menu" and works just like a menu in a roadside cafe. If you wanted, for instance, two eggs, fried potaoes, whole wheat toast, and coffee, you could order a number 6. To use the menu, just type the number you have chosen. Press the RETURN key to stop any demonstration and return to the menu.
You may notice that the first selection is handy for setting the color on your TV. If you choose this selection, some bars of light will appear on the screen. Turn off any automatic fine tuning on your TV and, with some adjustment of the Picture, Brightness, Color, and Hue knobs, these bars of light will become sixteen bars of color corresponding to the sixteen colors that can be used for programming the Apple. The names of the colors are abbreviated below the corresponding bars.

The second option is the same as the first one except that numbers from zero to fifteen have replaced the color names that appeared below the color bars. The Apple refers to the colors by these numbers.

You'll have to figure out the third and fourth options yourself. We don't think you'll have any trouble. Why don't you try one now?

COLOR SKETCH

Submitted by: Bruce Tognazzini
Program Language: Integer BASIC

If you like to draw, you'll enjoy Color Sketch. The Color Sketch program is fun for children and adults alike. With it, you can draw pictures on the Apple screen in fifteen different colors. Then, if you choose, you can save your pictures on cassette or diskette for future enjoyment.

The cursor, controlled via the game controls, leaves a trail of color behind it, except when the special "invisible" color is used. "Invisible" allows you to move the cursor around the screen without changing in your picture. The colors are all displayed below the sketching screen. If you wish to change the color you are using, simply type the letter that appears below the color of your choice. The current color's name is shown in the text window at the bottom of the screen.

All the instructions you need to draw colorful pictures with Color Sketch are included in the program. Just LOAD the tape labeled COLOR SKETCH and then type RUN . When your picture is finished, press the ESC key to access more Color Sketch program features. These extra features include the ability to LOAD pictures from tape or diskette and to SAVE pictures onto tape or diskette, a "back to the drawing board" feature, and more. All the extra features are as easy to use as typing a number.

When you have finished using the Color Sketch program, type CALL -340 if you wish to look at another program. The Color Sketch program stores information in areas of the Apple's memory in which programs are not usually stored. CALL -340 puts things back in the order the Apple expects them to be. Typing CTRL B from the monitor (asterisk prompt) will do the same thing, but, with CALL -340 , you don't have to enter the monitor first.

SUPER MATH

Submitted by: Ron Graff
Program Language: Integer BASIC

Kid: "Aw, gee, I don't like flashcards. Can't we do it later?"

Parent: "Come on, dear. If you had studied for your math test you wouldn't have to practice your arithmetic now."

Does this scene sound familiar? Flash cards can be uninteresting, but if you fall behind in math class . . .

Fortunately there is a solution to the flash card dilemma . . . the Super Math program. With your Apple and this program you can practice arithmetic to your heart's content (or your teacher's) without once glancing at a flash card.

To use the Super Math program put the tape labeled SUPER MATH into your recorder and LOAD it. Then type RUN . The program will first greet you and ask for your name. When it does, type your name and then press RETURN . Now you are ready to set up your lesson.

Several options are open to you at this point. The first one lets you choose the type of problems you would like to practice (addition, subtraction, multiplication, division). After that you can choose whether or not you would like to have one number consistent in all the problems for this lesson (handy for learning the multiplication tables). Then you get to choose the level of difficulty at which you wish to practice (from one to ninety-nine). If you aren't sure what level to choose, type Ø and the program will give you a series of test questions designed to determine the level that is right for you.

After you've chosen the level at which you wish to practice, the problems will appear on the screen one at a time. A "dash" appears where the answer will appear when you type it. The dash moves over as you type each digit of your answer. If you mistype something, you can use the arrow keys to edit your answer. When you think your answer is right, press RETURN and the program will check to see that it is. If your answer is incorrect, you will be given a chance to correct it, but the answer will be scored as wrong; if the answer you give is correct, the program will raise the problem level and let you choose whether you would like to do another problem, or whether you would like to score your progress so far. You can continue doing problems as long as you like.

That's all there is to using the Super Math program. Now that you know how to use it, you will never have to use flash cards to study arithmetic again.

APPLEVISION

Submitted by: Bob Bishop
Program Language: Integer BASIC

Bob Bishop did a wonderful job with Applevision. It is certainly one of the best examples of computer animation ever done on the Apple. Applevision is reminiscent of an old-fashioned vaudeville act, complete with music. So curl up by the fire with a big bowl of popcorn and the family gathered all 'round, to watch Applevision.

To see this marvelous piece of animation in action type HIMEM:16384 and then press RETURN . If you are an experienced programmer, you will know what this command means. Now LOAD the program from cassette tape, and then RUN it. If you LIST the program before you RUN it, you will see that it is not written in the BASIC you are used to. That is because the first part of this program is written in machine language. Not to worry. It will RUN just as well as a program written entirely in BASIC.

When you are finished watching the program, turn your Apple off, and then, if you wish to continue using it, turn the Apple back on again. This must be done because the Applevision program resets memory locations in the Apple so that, when it is finished, other programs will not RUN properly.

BIORHYTHM

Submitted by: Computer Generation
Program Language: Integer BASIC

The theory of biorhythms has been in the news in recent years. It states that human capacities vary in cycles. The most important cycles are these:

Physical

The physical biorhythm, varying over a 23-day cycle, is regarded as an indirect indicator of physical well-being, basic bodily functions, strength, coordination, and resistance to disease.

Emotional

The emotional biorhythm, varying over a 28-day cycle, is regarded as an indirect indicator of sensitivity, mental health, mood, and creativity.

Mental

The mental biorhythm, varying over a 33-day cycle, is regarded as an indirect indicator of alertness, analytical ability, and mental receptivity.

These biorhythms are thought to affect behavior. When they cross a "baseline" the functions change phase--become unstable--and this causes Critical Days. These days are, according to the theory, our weakest and most vulnerable times. Accidents and illnesses are thought most likely on physically critical days. Depression, quarrels, and frustration are thought most likely on emotionally critical days. Mental slowness, resistance to new situations, and unclear thinking are thought most likely on mentally critical days.

To get a color plot of your three biorhythms over a month-long period, follow these steps:

1. LOAD the program.
2. Type RUN and press RETURN .
3. When asked, type your name.
4. When asked, type your date of birth.
5. When asked, type the date for which you want a forecast.

The program will calculate the number of days between the birth date and the forecast date, and print it on the screen, then plot the three biorhythms in three different colors on the screen. It will then ask you whether you want another plot, or whether anyone else wants one. If you reply No to both questions, the program will end.

PINBALL

Submitted by: Charlie Kellner
Program Language: Integer BASIC

What do you do on a Friday evening when you've sprained your ankle and can't walk down to the neighborhood pizza parlor to play pinball? Let your fingers do the walking and play pinball with your Apple, of course! It's easy and fun, and you'll save plenty on shoe leather.

The Pinball program is a dynamic simulation of a real-life pinball machine, complete with bumpers, spinners, flags, and more. You move the "flipper" back and forth with one of the Apple's game controls. The program scores your game and keeps track of the highest scores for that game. This feature makes it easy to compete against yourself and others.

To play Pinball, LOAD the tape labeled PINBALL and then type RUN . Your Apple will give you playing instructions on request. You may never go to a pizza parlor again--except to get pizza.

INFINITE NUMBER OF MONKEYS

Submitted By: Bruce Tognazzini
Program Language: Integer BASIC

What high-placed, red-faced government officials would rather this story never got out? What federal agency did its best to suppress it and almost got away with it? What Washington paper thought the story just too hot to handle?

Now, for the first time ever, anywhere, APPLENEWS prints the TRUTH of the most mendacious mess of monkey-business since the Cooper-Simian scandals of 1883.

Apple's fearless reporter went where others with greater olfactory sensibilities feared to tread to bring back the greatest story of the 20th century. Find out why that New England paper company chopped down a 1000 year-old forest in Georgia. Learn the truth behind the recent Brazilian banana shortage.

Instructions

LOAD the program in Integer BASIC and RUN it; you will be presented with the menu. For a discussion of choices 2 and 3, see the Integer BASIC Subroutine Package, described on the next page. Type "1" and press RETURN , and the story will be told. Since the author can't imagine why anyone would wish to leave this program during its execution, the only way out before the end of the story (once it has begun) is to type a CTRL-C .

Our intrepid journalist made three trips to the historic site during the course of writing this program; three subjects were viewed, and three results were recorded. Unfortunately, at some point the manuscripts became jumbled, so do not be surprised to discover three different outcomes, chosen almost, it seems, at random.

After the story has been presented, you will be told that pressing RETURN will return you to the menu. Having returned, you may then exit the program by selecting choice 4.

If you have only 16K bytes of memory, you may reduce the program size by typing

```
DEL 7ØØ, 132Ø
```

before SAVEing it. This will delete the page-list program which is not used by The Infinite Number of Monkeys program itself.

INTEGER BASIC SUBROUTINES

Submitted By: Bruce Tognazzini
Program Language: Integer BASIC

The Infinite Number of Monkeys (see previous page) is an approximately 8K program which has been expanded to almost 2ØK by adding a large number of REM statements. These REM statements describe in detail the Integer BASIC Subroutine Package, a group of short but powerful subroutines and functions (many just a single line long) which expand the power of Integer BASIC. This program was also designed to be studied by those who are just learning BASIC. It assumes nothing beyond the user's having read the Apple II BASIC Programming Manual. It is structured in clearly defined subroutine blocks which interact as little as possible. This allows the user to lift, for example, the VAL(V) function, whole, out of this program and append it to any other program. (Please see the Programmer's Aid #1 manual, chapters one and two.) The author keeps a diskette of such routines that can be immediately EXECed into place (see the Apple Do's and Don'ts of DOS manual).

The Subroutine Package in The Infinite Number of Monkeys program contains the following subroutines and functions:

1) Automatic LOMEM:
2) Integer BASIC CHR$ Function
3) Moving Blocks of Memory
4) Pseudo Typewriter
5) Page LIST Program
6) Integer BASIC VAL Function
7) Illegal Statement Writer

Instructions

LOAD the Infinite Number of Monkeys program in Integer BASIC and RUN it. You will be presented with a menu. For a discussion of choice 1, see the discussion of The Infinite Number of Monkeys on the previous page.

If you have a 16K byte Apple, you may reduce the size of the Subroutine Package by typing

```
DEL 19ØØ, 28ØØ
```

This will eliminate the text portion of The Infinite Number of Monkeys while retaining the entire subroutine package.

A Note on Renumbering

Select choice 2, THE TABLE OF SUBROUTINES , from the main menu. Yes, it did happen. Return to the menu (by pressing the RETURN key) and select choice 2 again. When the author began this program, it was decided to make frequent, almost constant use of the Renumber/Append program, which you probably have on-board in Programmer's Aid #1, but which is also available in a softer form. To allow fluid renumbering, everything in this program, including the line numbers on this menu, must be updated when the program is renumbered. If you LIST line 184Ø, you will see the line that produced the menu; as the menu is a LISTing, rather than a print-out, the updating is done by the renumber program itself.

On Using the Auto-List Program

If you are not already there, please return to the main menu (by pressing the RETURN key) and select choice 3, THE AUTO-LIST PROGRAM . You will again be shown the Table of Subroutines and will be given instructions on the use of the auto-lister. This lister will show one text screen of lines at a time and then pause either until you are ready for the next page (press the space bar) or until you wish to cancel the listing and return to the Table of Subroutines (press ESC). You give Apple your list request by typing the word LIST followed by the number of the line with which you wish to start LISTing. (There is no need to specify an ending line as pressing ESC will end the listing at any time.) One of the delights of the program is that it allows such variants as LSIT and LIAR without giving you a SYNTAX ERR .

(Just to show how far some people will carry something, ESC to the Table of Subroutines and, after noting the line in the lister's instructions which says, TYPE "LIST 3Ø" AND PRESS "RETURN" , LIST beginning at line 728. This is how one gets an updated line reference <u>inside</u> a print statement! All this will be much more dramatic if you subject the program to renumbering; we suggest you try it. Please ESC back to the head of the list program.

An Exploration

The following documentation will provide an overview of the various blocks, starting from the beginning of the program, and give more in-depth information where appropriate. Please type LIST and look at the program's comments as you read those supplied here.

AUTOMATIC LOMEM:

Line 5Ø

Attempting to enter LOMEM: within an Integer BASIC program will produce a SYNTAX ERR ; line 5Ø allows you to accomplish the same task legally. The desired LOMEM: address in this case is 3Ø72; just replace 3Ø72 (in both places where it appears) with whatever LOMEM: your program requires. This must be done before any variables or arrays are either DIMentioned or assigned values, because the LOMEM: subroutine will clear the variable table when it is run. The second and third statements in the line will act as a CLR command when used alone.

Purpose: to set LOMEM: to a desired number from within a BASIC program.

To Use: Place at beginning of program and execute before initializing any variables.

Affect: equivalent to typing LOMEM: in immediate mode. Resets bottom of variable table and CLeaRs variable table.

CALLS: No other lines are referenced by this routine.

INTEGER BASIC CHR $ FUNCTION

LINES 11Ø, 12Ø

```
11Ø CHS = CHR +128* (CHR<128)

12Ø LC1= PEEK (224): LC2= PEEK (
    225)-(LC1>243): POKE 79+LC1-
    256*(LC2>127)+(LC2-255*(LC2>
    127))*256,CHS:CHR$="A":RETURN
```

The author was able to make free use of quotation marks throughout The Infinite Number of Monkeys program because of the inclusion of this routine. It is also the alleged generator of monkeytalk.

This subroutine gives you the same ability in Integer BASIC that the CHR$ function delivers in, for example, Applesoft BASIC.

In BASIC, the CHR$ function returns a character when given its numerical position in the list of ASCII characters. (The list may be found in the Applesoft Reference manual.) Many characters cannot be generated on the Apple II since its keyboard is upper-case only. Other characters, such as CTRL-C , cannot be stored in a program since they have special functions in the system. Yet these characters are often necessary when controlling external devices, writing programs that write programs, and in many other applications.

It is important that the second line of the routine be typed <u>exactly</u> as shown, with the exception that the line number may be changed. If an error is made, your program may be irreversibly altered. SAVE the program before RUNning, so should an error be discovered upon RUNning, you may reLOAD the program and correct the line. Once entered and checked, the program is completely safe and has the advantage over other Integer BASIC CHR$ schemes of being completely independent of the variable table. If the line is used in a long program on an Apple with more than 32K bytes of memory, there is a very remote possibility that one could get a *** >32767 ERR. If this should happen, insert a REM statement with about 8Ø characters on a higher line number to force the CHR$ function down in memory.

<u>Theory of Operation:</u> See Illegal Statement Writer

Purpose: to convert the ASCII code number placed in CHR into its equivalent character in CHR$

Setup: no initializing is necessary

To Use:

```
        Input--Is placed in CHR
               GOSUB 11Ø
        Output--Is found in CHR$
```

Variables Affected: CHS, LC1, LC2, CHR$

Calls: no other lines are referenced by this routine.

Example: contained on line 18Ø

MOVING BLOCKS OF MEMORY

LINES 33Ø,38Ø

You may move any block of memory to any other block by changing the addresses listed on line 33Ø, or replacing the variable names with your desired addresses. The author used these names to make the process as clear as possible. ("DESTINACION" is purposely mispelled: "DESTINATION" contains the reserved word, "AT".) Be sure you do not move a block of memory on top of your program.

Purpose: Move a block of memory to another position in memory.
Equivalent to Monitor move routine.

Setup: No initializing is necessary

To Use:

```
        Input--None
               GOSUB 33Ø
        Output--None
```

Variables Affected: DESTINACION, SOURCESTART, SOURCEFINISH

Calls: No other lines are referenced by this routine.

PSEUDO TYPEWRITER

LINES 3ØØ, 69Ø

This is a very handy package that will take loosely entered PRINT statements and "type" them out to the screen, breaking lines between words, with options for typewriter-like sound and hesitation. With minor modification, as described below, it can be used for higher speed word-at-a-time output. The author developed it because he found he was spending an inordinate amount of time carefully formatting PRINT statements in his programs, only to find himself doing it all over every time he changed one word. With this system, editing is quite reasonable.

Theory of Operation

The programmer supplies the routine with a sentence or phrase contained in S$. Starting at the head of the main loop (Line 6ØØ), the first word is extracted from S$ and placed in W$. (Lines 61Ø, 63Ø). Line 65Ø adds the number of characters in the word to the current TAB position and, if the word will not fit on the line, GOSUBs the scrolling routine. Lines 66Ø and 67Ø print out the letters in the word, one-by-one, adding sound and delay if requested by main program.

DELeting line 66Ø and replacing it with: "66Ø POKE 5Ø,63: PRINT WRD$;: POKE 5Ø,255" will print one whole word at a time. (Good programming style dictates that the system should be left in the most normal condition possible when a user aborts a program. By turning off white text mode as soon as possible, the user is unlikely to be caught in reverse-mode after typing a CTRL-C .

Note on Line 54Ø: when writing subroutine blocks that could be initiallized more than once, use a flag that is turned on when your variables are DIMensioned. This flag can then prevent their being DIMed again. This will prevent any REDIM'ED ARRAY ERR messages.

Purpose: To print-out properly formatted text on a white background from unformatted PRINT statements.

Setup: Initialize by a GOSUB 53Ø. This will DIM the variables and run a short routine immitating paper being scrolled into a typewriter.

To Type Words:

```
              Input--S$ should contain desired word(s), phrase,
                     or sentence
                     GOSUB 6ØØ
              Output--all output is to screen
```

To Manually Scroll: (for paragraphing, etc.)

```
                    Input--none
                    GOSUB 57Ø
                    Output--all output is to screen
```

User Selectable Flags:

```
                       Set SOUND equal to 1 for typewriter sound
                       Set DELAY equal to a number between Ø (no delay)
                       and 5Ø (long delay) for hesitation between
                       characters
```

Variables Affected: S$, SOUND, DELAY, K, S, SS, WRD$, TIME, CURRENTPOS, SOURCESTART, SOURCEFINISH, DESTINACION

Calls: no routines outside Lines 3ØØ, 69Ø are CALLed, but AUTO-LOMEM: (line 5Ø) must be executed before any variables are initiallized to allow use of Page 2. It is useful to retain Lines 155Ø to 166Ø to keep the directions. All other lines may be deleted to isolate the package for use in your own programs.

Example: LIST and RUN Line 164Ø.

This package (with some of the REM statements stripped out) is very small and extremely easy to use, with only one variable, two optional flags, and three calls. It can save you hours of frustrating work; it took the author less than 2Ø minutes to enter the complete text for the (pre-written) story of The Infinite Number of Monkeys. Subsequent editing was done with a minimum of bother.

To edit any PRINT statements on the Apple in either Integer BASIC or Applesoft BASIC, first type ESC @ (press and release the ESC key, then type @) to clear the screen, then type POKE 33, 33 .

Now LIST the line(s) you wish to edit. This reduced text window will eliminate the extra spaces in the listed lines and thus, the need for typing ESC A 's to skip over them. When you are through editing, type TEXT to restore the normal text window.

PAGE LIST PROGRAM

LINES 7ØØ, 132Ø

Because The Infinite Number of Monkeys was written as a tutorial, the author has included this listing routine to make the whole program as easy as possible to study. Since it is written in BASIC, it is fairly slow. But it does its job, and includes one extremely interesting line, "LIST X,Y" (Line 1Ø3Ø -- try typing it in yourself). This line, and a one-line routine for entering such lines, will be discussed in detail in Illegal Statement Writer, the last section of this discussion.

Theory of Operation

When the user types a list request, such as, "   LIST 1Ø4Ø", the program carries out the following operations:

1) Line 78Ø. Leading spaces are removed, leaving, "LIST 1Ø4Ø".
2) Lines 82Ø, 83Ø. The first four characters are matched against the command table. If a valid command is found, the command is removed from the string, leaving , " 1Ø4Ø".
3) Next, the string is processed by the VAL(V) FUNCTION (see below), which returns with the integer number 1Ø4Ø contained in V.
4) Finally, a loop X is started from 1Ø4Ø to 32767 and the lines are listed out. See the REM statements contained in the LIST program for specifics.

Some of the newer programmers may feel the author went to a lot of unnecessary work just so the user would have to type "LIST" instead of simply entering the line number. More experienced programmers, of course, have no doubt he went to a lot of unnecessary work. The author's rather hastily constructed excuse for this was that he had always wanted to parse somebody's syntax. He is being carefully watched.

INTEGER BASIC VAL FUNCTION

Lines 14ØØ, 154Ø

One of the marks of a professional programming job is that the program doesn't "blow-up" when the user makes a predictable error. For example, when The Infinite Number of Monkeys is first RUN, it is not inconceivable that a naive user could type "END" instead of "4". Of course, the author has been very careful to explain what kind of input the computer wishes, but it is still good programming practice to avoid the avoidable. RUN The Infinite Number of Monkeys, and, at the menu, type END instead of 4.

Two events took place. First, the computer simply did not accept your input, giving clear indication that you did not enter it correctly. Second, in a brief flash, it was announced, "EXTRA IGNORED". That was the VAL(V) FUNCTION talking. When you input any number in this program, you are entering it into a string. That string is then processed by the VAL(V) FUNCTION and, if a valid number between -32767 and 32767 is found, that number is returned in an integer variable for use by the program. This gives the programmer complete control over what is coming in; the user cannot, short of intentional sabotage, crash the program. It is a simple routine to use and is easy installed within your program. The "EXTRA IGNORED" message alerts users either that they have entered a number larger than 32767 and the extra digits have been rejected, or that they have entered non-numeric information following the number. This feature may be eliminated from the function if not needed. In fact, it is not needed in this program, but was included to allow you the option within your own programs.

Still at the main menu, try entering " 3ABC DE". The VAL function ignores the leading spaces, and evaluates the initial 3 as your response. The extra is ignored and execution of the command to go to the list routine takes place. You may now LIST the VAL(V) function, if you wish, and see how it is constructed. Line 141Ø will only DIM V$ if it has not been previously DIMentioned, line 142Ø will cause a return if V$ is null (empty). Therefore, to initialize this routine the programmer executes a "GOSUB 14ØØ". The next time a "GOSUB 14ØØ" is ordered, the subroutine will expect V$ to contain the number to be processed.

Purpose: This routine converts the string V$ into an integer value V

Setup: Initialize by a "GOSUB 14ØØ" before using V$

To Use:
```
        Input--desired input is placed in V$
               GOSUB 14ØØ
        Output--integer output is found in V upon return
```

Variables Affected: V$, V, VV, VVV, MINUSFLAG

Calls: no other routines are referenced by the VAL(V) function

Example: LIST and RUN line 139Ø

The Infinite Number of Monkeys itself, from line 19ØØ to line 28ØØ, shall be left to the reader to explore. It is not particularly distinguished -- it was not meant to be. It is a sea of unformatted text and GOSUB statements. There are a fair number of delay loops that the author used in timing, or punctuating, the animated text; there is the section from line 238Ø to 267Ø that contains the "monkey business".

ILLEGAL STATEMENT WRITER

Line 1Ø2Ø (Called by 1Ø15, acts upon 1Ø3Ø)

```
1Ø2Ø LC1= PEEK (224):LC2= PEEK(
     225)-(LC1>243): POKE 81+POS+
     LC1-256*(LC2>127)+(LC2-255*
     (LC2>127))*256,CMD: RETURN
```

A great feature of Apple Integer BASIC is its entry-time syntax checking. You need not wait until RUN-time to find out the depths of your folly; Apple will beep it to the world just as soon as you press the RETURN key. While this is a generally commendable feature, it does inhibit exploration of some potentially interesting lines by steadfastly refusing to accept what seem to be perfectly reasonable commands. Surely, there should be some room for legitimate differences of opinion, Apple!

"BEEP! *** SYNTAX ERR."

Why are we not permitted to find out what happens when line 5Ø says, "5Ø NEW" and we RUN 5Ø? Why can't we DELete line 123 after it has already been used? Why are we forbidden to LIST X?

"BEEP! *** SYNTAX ERR."

What we need is the Illegal Statement Writer. With this handy little 1-line subroutine, you can write anything you want! (Of course, poor Apple may be a litte confused over such lines as, "1ØØ A$=27/PRINT", but you can write it.)

The Illegal Statement Writer was created to allow BASIC programmers to POKE normally rejected commands, characters, and numbers into their programs. Apple is quite capable of LISTing from a variable to a variable ("1Ø3Ø LIST X,Y") if one can just get the line into memory. Apple is perfectly happy with line numbers higher than 32767 if one can just get them entered. (Most of you will have seen line 65535 as the last line of many programs.) One can even poke quotation marks (ASCII 162) inside a quote! This subroutine gives everyone the power that has been reserved for the machine-language jockeys until now.

Theory of Operation

The following paragraph is a highly technical explanation of how the subroutine works and need not be understood or even read to make full and complete use of the subroutine. The analysis is presented for advanced programmers who may find the previously undocumented pointers identified herein useful in designing new subroutines in this family. The author has written 5 different types of routines, including the CHR$ function in this program, all based on being able to pinpoint the actual memory location of a BASIC program line during RUN-time. It is his belief that there are many more to be discovered.

When Integer BASIC encounters a statement such as PRINT X , it reads it, parses (interprets) it, and goes off to execute it. When it departs, it stores its current program position in memory locations 224 and 225. Upon completing execution of the statement, it reads these locations to know where to resume. This subroutine PEEKs these locations to pinpoint its own actual location in memory, and with that information, is able to POKE your chosen command (CMD) into a position (POS) in the next line. LC1 (location1) is given the low byte of the address of the colon which immediately follows it, as this is where the program return pointer is when 224 is PEEKed. (For an explanation of high and low bytes, see Instructions For Use below.) LC2 is likewise given the high order byte of the address of its following colon. If LC1 was greater than 243, LC2 will have "clocked over" by the time the pointer reaches it, and thus a 1 is subtracted if LC1>243. We are left then with the actual location of the first colon stored in LC1 and LC2. 82 bytes beyond this colon is the command LIST in line 1Ø3Ø. If we make POS = 1 and POKE 81 + POS + <the location in memory> , <number of desired token>, we can POKE any command token into whatever place we want. (The balance of line 1Ø2Ø that deals with adding and subtracting 255's and 256's turns any location numbers higher than 32767 into 2's complement form so users with more than 32K can use the routine.)

The only place this routine will not work properly is just above 32767; if you have more than 32K of memory and encounter a *** >32767 ERR, enter a REM statement with 1ØØ spaces or so on a higher line number to force the Illegal Statement Writer down in memory. The REM statement may be removed after you are through POKEing your line.

Instructions For Use

First, type:

```
DEL Ø, 1Ø14
DEL 1Ø31, 32767
```

This will leave just three lines. The first line, 1Ø15, controls the subroutine, telling it what to POKE where. The second line is the POKEr, the third, the POKEe.

* * * NOTE * * *

Line 1Ø2Ø is to be retained, unmodified, in each example below.

Retype line 1Ø3Ø to read:

```
1Ø3Ø PRINT X,Y
```

Then, RUN. It should read, 1Ø3Ø LIST X,Y again. Apple Integer BASIC and Applesoft are partially compiled languages. When you type a command such as GOTO into a program, Apple substitutes the four characters G-O-T-O with a "token" (a number between Ø and 127 which represents a command). In the case of GOTO, this token is the number 95. When you LIST the program and Apple encounters a 95 it looks up this token in a table, finds out it means GOTO, and then prints it out that way. This compilation makes your program smaller and faster.

Tokens

DEL 1Ø15 and enter the following:

```
1Ø3Ø PRINT
1Ø POS=1: FOR CMD = Ø TO 127: GOSUB 1Ø2Ø: PRINT CMD,: LIST 1Ø3Ø :NEXT
CMD:END
```

Then RUN. You have just LISTed the complete token table. Most tokens can be used legally in differed mode (written into a program); below is the table of Integer BASIC tokens. (The HEX numbers are for the benefit of the aforementioned machine-language jockeys and are to be ignored by those of us of a more civilized ilk.)

Integer BASIC Token Table

NUMBER DEC	HEX	TOKEN	COMMENTS
Ø	$Ø	HIMEM:	Token irrelevent - used internally as begin-of-line.
1	$1		End-of-line token - each line ends with a 1
2	$2	_	Used internally in delete line processing
3	$3	:	Colon for statement separation
4	$4	LOAD	
5	$5	SAVE	
6	$6	CON	
7	$7	RUN	RUN n, where n is a line number
8	$8	RUN	RUN from first line of program
9	$9	DEL	
1Ø	$A	,	Comma used with DEL (DEL Ø,1Ø)
11	$B	NEW	
12	$C	CLR	
13	$D	AUTO	
14	$E	,	Comma used with AUTO (AUTO 1Ø,2Ø)
15	$F	MAN	
16	$1Ø	HIMEM:	The real thing
17	$11	LOMEM:	

The following are numeric operators

18	$12	+	
19	$13	-	The associated parentheses are 56 and 114
2Ø	$14	*	Example: A = 14 * (27 + 15)
21	$15	/	

The following are numeric variable logical operators

22	$16	=	Example: IF X = 13 THEN END
23	$17	#	
24	$18	>=	
25	$19	>	
26	$1A	<=	
27	$1B	<>	
28	$1C	<	
29	$1D	AND	
3Ø	$1E	OR	
31	$1F	MOD	
32	$2Ø	^	
33	$21	+	Unused
34	$22	(	Used in string DIMs: DIM A$(n)
35	$23	,	
36	$24	THEN	Followed by a line number: IF X = 3 THEN 1Ø
37	$25	THEN	Followed by a statement: IF X = 3 THEN A$ = "CAT"
38	$26	,	Used with string inputs: INPUT "WHO", W$
39	$27	,	Used with numeric inputs: INPUT "QUANTITY",Q
4Ø	$28	"	Beginning quote
41	$29	"	Ending quote
42	$2A	(	Substring left parenthesis: PRINT A$(12,14 used with 114 as right parenthesis (see also 66)
43	$2B	!	Unused
44	$2C	!	Unused
45	$2D	(	Variable array left parenthesis: X(12) used with 114 as right parenthesis
46	$2E	PEEK	Uses 65 and 114 for parentheses
47	$2F	RND	" "
48	$3Ø	SGN	" "
49	$31	ABS	" "
5Ø	$32	PDL	" "
51	$33	RNDX	Unused
52	$34	(	Used in variable DIMS: DIM A(1Ø)
53	$35	+	Unary signum: A = +5
54	$36	-	Unary signum: B = -5
55	$37	NOT	Numeric
56	$38	(	Used with 114 in logic statements and numeric operations: IF C AND (A = 14 OR B = 12) THEN X = (27 + 3)/ 13
57	$39	=	String logical operator: IF A$ = "CAT" THEN...
58	$3A	#	String logical operator
59	$3B	LEN(	Uses 114 as right parenthesis
6Ø	$3C	ASC(	" "
61	$3D	SCRN(	" "
62	$3E	,	Comma used with scrn: PRINT SCRN(X, Y)
63	$3F	(	Used with 114 after PEEK, SGN, ABS, and PDL 64 $4Ø
$		String	
65	$41	$	Unused

66	$42	(	Special case string array right parenthesis. used when string array is the <u>destination</u> of the data. in the example, A$(1) = B$(1) , the A$ left parenthesis will be 66 and B$'s will be 42. used with 114 as right parenthesis
67	$43	,	DIM <varname>, <string varname>
68	$44	,	DIM <varname>, <numeric varname>
69	$45	;	String PRINTs: PRINT <varname>; <string varname>; "X"
7Ø	$46	;	Numeric PRINTS: PRINT <varname>; <numeric varname>; 7
71	$47	;	End of PRINT statement: PRINT A;
72	$48	,	String PRINTS: PRINT <varname>, <string varname>, "X"
73	$49	,	Numeric PRINTS: PRINT <varname>, <numeric varname>, 7
74	$4A	,	End of PRINT statement: PRINT A$,
75	$4B	TEXT	
76	$4C	GR	
77	$4D	CALL	
78	$4E	DIM	String var. Parentheses 34 and 114, comma 67
79	$4F	DIM	Numeric var. Parentheses 52 and 114, comma 68
8Ø	$5Ø	TAB	
81	$51	END	
82	$52	INPUT	String with no prompt: INPUT A$
83	$53	INPUT	String or numeric with prompt: INPUT "WHO", A$ uses comma 38 INPUT "NUMBER", A uses comma 39
84	$54	INPUT	Numeric with no prompt: INPUT A

The following are for FOR/NEXT loops

85	$55	FOR	
86	$56	=	
87	$57	TO	
88	$58	STEP	
89	$59	NEXT	
9Ø	$5A	,	
91	$5B	RETURN	
92	$5C	GOSUB	
93	$5D	REM	
94	$5E	LET	
95	$5F	GOTO	
96	$6Ø	IF	
97	$61	PRINT	String variable or literal: PRINT A$: PRINT "HELLO"
98	$62	PRINT	Numeric variable: PRINT A
99	$63	PRINT	Dummy PRINT: PRINT : PRINT

1ØØ	$64	POKE	
1Ø1	$65	,	Comma used with POKE
1Ø2	$66	COLOR=	
1Ø3	$67	PLOT	
1Ø4	$68	,	Comma used with PLOT
1Ø5	$69	HLIN	
1Ø6	$6A	,	Comma used with HLIN
1Ø7	$6B	AT	AT used with HLIN
1Ø8	$6C	VLIN	
1Ø9	$6D	,	Comma used with VLIN
11Ø	$6E	AT	AT used with VLIN
111	$6F	VTAB	
112	$7Ø	=	String -- non-conditional: A$ = "HELLO"
113	$71	=	Numeric -- non-conditional: A = 14
114	$72	)	The only right parenthesis token -- won most-popular-token award in Atlantic City
115	$73	)	Unused
116	$74	LIST	LIST a range of numbers or specific number: LIST 1Ø : LIST 12Ø, 32767
117	$75	,	Comma used with LIST
118	$76	LIST	LIST entire program
119	$77	POP	
12Ø	$78	NODSP	String variable
212	$79	NODSP	Numeric variable
122	$7A	NOTRACE	
123	$7B	DSP	String variable
124	$7C	DSP	Numeric variable
125	$7D	TRACE	
126	$7E	PR#	
127	$7F	IN#	

To use an illegal token inside a program, there must first be a legal line in which to POKE the new token. Because we could not enter, "LIST X,Y", we entered "PRINT X,Y" and then changed the two tokens. If you wish to have, "1Ø3Ø DEL 1Ø", then first enter "1Ø3Ø PRINT 1Ø" or "1Ø3Ø INPUT 1Ø". Then POKE in the new token.

<u>Variable Names and Spaces</u>

Variable names are made up of an alpha character which may be followed by a series of alpha or numeric characters. <u>Anything</u> other than alphanumeric characters appearing outside of quotes and REM statements are tokens.

Spaces between tokens, numbers, and variable names are deleted upon entry and re-supplied during LISTing. In counting the number of bytes in a line, all spaces outside of quotes and REM statements should be overlooked. When a REM is LISTed, one space is inserted after the word REM; therefore, the statement which lists as REM APPLE uses 6 bytes of memory, not 7.

Characters

The numbers from 128 to 256 are ASCII characters in Integer BASIC. A chart of these characters can be found in many computer manuals, including the Applesoft Reference manual. If your chart lists characters with numbers from Ø to 127, just add 128 to compute their "negative-ASCII" counterparts. (You might find it useful to write the higher numbers into your Applesoft Reference manual.) Enter the following lines:

```
1Ø3Ø PRINT "A"
1Ø POS =3: FOR CMD=128 TO 255: GOSUB 1Ø2Ø: PRINT CMD,: LIST 1Ø3Ø: NEXT
CMD: END
```

Then RUN it. To POKE the ASCII numbers between the quotation marks, it was first necessary to set POS. As PRINT is a command word, and therefore a single token, or byte, and its position is 1, then the space between the quotes is 3. By counting out from the beginning of the line, you may POKE your command, character, or number anywhere within the line.

Numbers

Numbers, like tokens, are converted upon entry. Unlike tokens, converted numbers always occupy 3 bytes. (The numbers we are considering are not numeric characters that make up part of a variable name, such as ALPHA3, but rather integers, as in X = 32 or ALPHA (3).)

The first byte of a line number contains the number of bytes in the line; the first byte of a number within a line contains as a flag the ASCII value of the first digit in the number. This tells the language that the following two bytes are a number. (The flag can be the ASCII value of _any_ digit -- 176 does nicely -- it is being used as a flag, not a value.) The actual number itself is made up of two bytes, the first being the low-order byte, the second, the high-order byte. When you enter a line number such as 1Ø3Ø into Apple, the language first computes 1Ø3Ø MOD 256 and puts that number in the first byte and then computes 1Ø3Ø / 256 and puts that number in the second byte. With two bytes, each capable of storing numbers from Ø to 255, the square of 256, or 65536 numbers may be expressed. (This 65K range is from -32767 to +32767, or from Ø to 65535 depending on its interpretation.)

Therefore, whenever you wish to POKE a number into a position past integers in the line, POS must be increased by exactly 3 for each integer to be leap-frogged, be that integer Ø or 32767. Try the following new lines, retaining line 1Ø2Ø:

```
1Ø3Ø X=14: PRINT "SAY @APPLE@" :END
1Ø POS = 13:CMD = 162:GOSUB 1Ø2Ø:LIST 1Ø3Ø:END
```

Then RUN it. You'll note, we've a quote within a quote. Your job is to put a quote at the end of the line, where the second @ is located. (The @ has been used arbitrarily; it could be any character.) Keep in mind that position 1 is where X is, that the reserved word (command) PRINT and the reserved word : (statement separator) are each one byte, and that the number 14 is three bytes. Any spaces outside of the quotes do not count. The only thing you must change is POS, the ASCII for a quotation mark, 162 stays the same. After you have both quotes, RUN 1Ø3Ø.

Finally, let's change line 1Ø3Ø to 65535. The low and high bytes of 65535 are both 255. The line number bytes are the two immediately preceeding POSition 1 in our line, thus they are -1 (the low-order) and Ø (the high-order). To change our line, type this new line 1Ø:

```
1Ø POS=-1: CMD=255: GOSUB 1Ø2Ø: POS=Ø:GOSUB 1Ø2Ø:LIST:END
```

Changing the line back to 1Ø3Ø will be left to you; the information on computing the byte values may be found in the second paragraph of this section on numbers. Just do exactly what the computer normally does.

Self-Writing Programs

It is possible to write a program which can write itself, using this line, by dropping HIMEM: down, POKING in syntactically correct lines off the top end of the program, and POKING a new HIMEM: into place when done using the following line:

```
<linenumber> HMM= 82 + POS + LC1 + LC2*256 : POKE 76, HMM MOD 256: POKE
77, HMM / 256
```

HIMEM: is always 1 past the end of the program; thus we add 82, not 81. The above line assumes that you will not write above 32767 in memory; be sure to bring HIMEM: sufficiently below this figure before beginning. One use of this method would be to write a program that would convert machine language into POKE statements inside your BASIC program; another would be a DATA program that would write: <linenumber> <your input data> : RETURN. (This one would undoubtably lead to another round of HANGMAN games.) Your imagination can supply you with many other uses.

Try this simple sample primitive program, after setting HIMEM: to (arbitrarily, but safely) 8192 and DELeting all but line 1020. (You need not enter the REM statements.)

```
10  DIM Y(13):REM OUR LINE WILL BE 13 BYTES LONG
20  Y(1)=0:Y(2)=80:Y(3)=97:Y(4)=40:Y(5)=193:Y(6)=208:Y(7)=208:
    Y(8)=204:REM TOKEN AND ASCII CODE FOR EACH BYTE IN LINE
30  Y(9)=197:Y(10)=41:Y(11)=3:Y(12)=81:Y(13)=1:Y(0)=14:REM THE
    FIRST BYTE IN THE LINE MUST ALWAYS CONTAIN THE LENGTH OF
    THE LINE
40  FOR POS = -2 TO 11:REM FIRST BYTE IN LINE IS AT POSITION -2
50  CMD = Y(POS +2):REM TO READ Y FROM 0 TO 13, WE MUST ADD 2
    TO POS
60  GOSUB 1020:REM POKE THE BYTE IN PLACE
70  NEXT POS: POS = POS-1 :REM LOOP UNTIL DONE
80  REM WHEN EXITING THE LOOP, POS WILL HAVE BEEN INCREMENTED
    TO 12, ONE GREATER THAN END-OF-LINE, SO SUBTRACT 1 TO MAKE
    IT ACCURATE
90  HMM=82 + POS + LC1 + LC2*256 : POKE 76,HMM MOD 256:
    POKE 77,HMM/256
100  LIST :GOTO 20480
```

Summary

It is hoped that the Illegal Statement Writer and the above discourse will lead you to a fuller understanding of Integer BASIC and computer language processes in general. While many illegal statements are fun to play with, "tricky" programming, such as having lines DELete themselves during run-time should be avoided in serious programming whenever possible. There are also many tasks that can be accomplished without using the command word itself (See AUTO-LOMEM:), allowing editing after entry. But when you need the Writer for something special, it'll be there, and you'll never have to take *** SYNTAX ERR as the final word again. (BEEP!)

SPACE WAR

Submitted by: D. Redington
Program Language: Integer BASIC

The dreaded Space War has begun. Our space fleet needs your strength and skill to defeat the Imperial Forces. If you choose to join us in our fight to preserve the freedom of space, you will be given a fighter ship. Your mission will be to destroy any enemy fighter ships that come in your range.

To play Space War, first LOAD the tape marked SPACE WAR . Then type RUN . The program will ask for your name and then assign your rank. When you begin the game, your starting rank will be Space Pilot Trainee. A target screen appears on your screen when the battle begins. You must maneuver your ship with the game controls until the enemy is in your sights. The buttons on the two game controls operate your weapons. One game control fires lasers and the other fires torpedos. The torpedo uses two units of energy per shot. The laser uses only one unit of energy per shot, but requires a more exact aim. The enemy fighters are clever, and it may be difficult to destroy them, but don't be discouraged. You will be amply rewarded for fighting well. If your performance in battle warrants it, your rank will be increased. With some practice, you may even make General.

When you have finished playing, type CALL -340 if you wish to look at other programs. The Space War program, like the Color Sketch program, stores information in areas of the Apple's memory where the Apple doesn't expect them. CALL -340 restores memory to the configuration the Apple expects.

Have a good game, and good luck!

APPLE TREK

Submitted By: Wendell Sander
Program Language: Integer BASIC

Stardate 3424.∅
The Free Galactic Federation has been invaded by battle cruisers of the Klarnon Empire. Several Federation star-systems are under attack. As an experienced officer, you have just been given command of the starship Endeavor. Your three-year mission, should you choose to accept it, is to search the Galaxy and destroy all Klarnon battle cruisers. If you succeed, the federation will survive. If not,.......

Your Apple II computer serves as a command console. IT informins you of operating conditions, movements and actions of enemy vessels, and carries out your orders.

The Endeavor is a complex vessel. You should take a few practice runs to learn how to operate it before you actually embark on your mission. Remember, the fate of the galaxy is in your hands.

LOAD the tape labeled APPLE-TREK in the usual manner, and then type RUN . The program will announce itself and the game setup will be revealed to you. Before making any decisions concerning the game setup, you should read further.

The Galaxy

The Galaxy in which the action takes place is subdivided into 64 quadrants of space, each of which contains 64 sectors. One sector is the smallest measured unit of space and may be occupied by any of the following: a star, a Klarnon, a starbase, or the Endeavor. Each quadrant is charted as an 8 by 8 array of sectors. Each sector within the quadrant is identified by coordinates as shown in Figure 1. Similarly, the Galaxy is charted as an 8 by 8 array of quadrants. Each quadrant within the galaxy is identified by coordinates as shown in Figure 1. The first coordinate of each pair gives the location along the Galactic North-South axis; the second, along the Galactic East-West axis.

1-1	1-2	1-3	1-4	1-5	1-6	1-7	1-8
2-1	2-2	2-3	2-4	2-5	2-6	2-7	2-8
3-1	3-2	3-3	3-4	3-5	3-6	3-7	3-8
4-1	4-2	4-3	4-4	4-5	4-6	4-7	4-8
5-1	5-2	5-3	5-4	5-5	5-6	5-7	5-8
6-1	6-2	6-3	6-4	6-5	6-6	6-7	6-8
7-1	7-2	7-3	7-4	7-5	7-6	7-7	7-8
8-1	8-2	8-3	8-4	8-5	8-6	8-7	8-8

Figure 1
Chart of Quadrants in the Galaxy or Sectors in a Quadrant

As discovered by the twenty-fourth century cosmologist Liebowitz, the Galaxy is a closed space, which is represented by the chart of quadrants. The Galaxy has no bounds: each edge of the chart is physically adjacent to the opposite edge of the chart so that quadrant 7-1 is bounded on one side by quadrant 7-2 and on the other side by quadrant 7-8. Likewise, quadrant 1-1 is bounded by quadrant 2-1 and 8-1. This concept is illustrated in Figure 2 below. In the "Four-Corners" region of the Galaxy, both axes "wrap around" at one point. (Mathematicians may find it helpful to model the Galaxy as a torus.)

7-7	7-8	7-1	7-2
8-7	8-8	8-1	8-2
1-7	1-8	1-1	1-2
2-7	2-8	2-1	2-2

Figure 2
Map of "Four-Corners" Region of the Galaxy,
Showing Adjacent Quadrants

The Starship Endeavor

The Endeavor is a powerful craft with somewhat more firepower and energy capacity than the Battle Cruisers of the Klarnon Fleet. However, the Endeavor is usually heavily outnumbered and is easily destroyed unless good maneuvering and firing strategies are employed.

Energy

At the beginning of your mission, the Endeavor will be stocked with 1Ø,ØØØ units of energy. Of these 1Ø,ØØØ units, 25ØØ units will be allocated to available energy stores, 25ØØ units to shield energy, and 5ØØØ units to photon torpedoes (1Ø torpedoes of 5ØØ units each). 1Ø,ØØØ units is the ship's maximum energy capacity.

Each of the three forms of energy the Endeavor uses has a specific application. The available energy is used for phasers, for propulsion, and for the ship's subsystems, one of which is the indispensible Apple 81 computer. The shield energy works to absorb enemy phaser and photon torpedo attacks. Without it the ship would be lost. The photon torpedoes, of course, are used to attack the enemy.

Each type of energy is adapted only to certain applications, but, fortunately energy of one form can be converted to energy of another. The shield energy is depleted with each enemy attack that scores a "hit." If the shield energy is allowed to get too low, the Endeavor may sustain damage to its subsystems.

Fortunately you, as captain, can transfere energy between the available energy stores and the shield energy. However such transfers take time and must be planned with caution.

Similarly photon torpedoes can be LOADed (formed from available energy) or UNLOADed (converted to available energy) depending on your energy needs. Both these transformations consume energy, however.

Energy Sources

When the Endeavor's energy level gets low, you will wish to restock. The ship's energy is restocked in two different ways.

One way is through the store of on-board dilithium crystals which generate energy continuosly at a rate of 5Ø units per Ø.1 stardate. The energy limit of 1Ø,ØØØ units cannot be exceeded, however, and any excess will dissipate before it can be used.

The other way to restock the Endeavor's energy is to stop at a restocking base. These bases are located at various points in the galaxy. A rendezvous with a restocking base brings the Endeavor back to maximum energy and repairs any subsystems that may be damaged, but also depletes all the base's energy. Each restocking base can be used only once.

Firepower

The Endeavor has two main forms of weaponry: photon torpedoes and phasers. Both these weapons are very powerful and each of them has a different application.

The Endeavor usually carries 1Ø photon torpedoes of 5ØØ energy units each. Photon torpedoes can be used to attack any point within a quadrant and can be fired in a straight line at any angle. If the photon torpedo hits a Klarnon ship or a star before reaching the edge of the quadrant, it will inflict 5ØØ energy units of damage. This will reduce the energy of a Klarnon by 5ØØ units or destroy a star.

Phasers are a focused energy weapon and are not blocked by stars or Klarnons. The phaser energy fired is divided equally among the targets selected and is reduced by the distance of the target. Phaser energy is drawn from the Endeavor's available energy supply. The number of phaser energy units fired is chosen by you. A phaser striking a Klarnon ship reduces the Klarnon energy supply by the amount of energy that hits the ship.

There is one form of weaponry that should only be used as a last resort. At the beginning of your mission, you will be asked for a SELF-DESTRUCT password. If you give the SELF-DESTRUCT command in conjunction with the SELF-DESTRUCT password the Endeavor will be destroyed with all hands lost. If sufficient energy is available, all enemy vessels in the quadrant will also be destroyed. This command has dire consequences, but it may be necessary in order to save the Federation.

Propulsion

The Endeavor has two forms of propulsion available: Warp Drive and Ion Drive. Warp Drive is used exclusively for movement from quadrant to quadrant. Ion drive is for short distance moves and can be used either for movement within one quadrant or for movement from one quadrant into an adjacent quadrant.

Under Ion Drive, the Endeavor moves 1 sector unit of distance per Ø.1 stardate. The direction of movement is determined by the COURSE, from Ø to 359 degrees. The distance is determined by the DURATION, from 1 to 9 sector units. The captain can set a course that will take the ship across the boundary of a quadrant and into the next one: energy and time will be consumed at the same rate as if all motion were in one quadrant. For this reason, it is usually quicker to move to an adjacent quadrant by ion drive, especially if the ship is near the boundary. The energy consumed by an ion drive move is 2Ø units plus 2Ø units per sector unit moved. The time consumed is Ø.Ø1 stardates per sector unit.

When using Warp Drive, the direction of motion is determined by the COURSE, from 1 to 359 degrees. The distance is determined by the WARP FACTOR, from 1 to 6 quadrant units. For the shortest possible move between two adjacent quadrants, moving along an axis at a COURSE of Ø, 9Ø, 18Ø, or 27Ø, the WARP FACTOR is 1. For a longer move along an axis, the WARP FACTOR is the number of quadrants moved. Angular moves require larger WARP FACTORS, as the Endeavor is moving along both axes at once: A diagonal move, on a course of 45, 135, 225, or 315, requires a WARP FACTOR of 1.4 times the number of quadrants moved, rounded to the nearest unit. The maximum value for the WARP FACTOR is 6, which moves the Endeavor 4 units north-south and 4 units east-west. An object within 3 sector units of the Endeavor in the direction of warp drive movement will block the movement. The energy required for a warp drive movement is 12 times the cube of the WARP FACTOR and is much larger for a large WARP FACTOR than a small WARP FACTOR. The time consumed by a warp move of any distance is Ø.1 stardate.

The Apple 81 Computer

The Endeavor is equipped with an Apple 81 computer which is at your disposal. The computer gives the Endeavor a major advantage over its Klarnon opponents. It is virtually impossible to defeat a squadron of Klarnons without the help of the computer.

The computer permits the Endeavor to lock photon torpedoes or phasers on selected Klarnon targets, to move while firing, and to compute angles for movement and for manual photon torpedo fire.

Klarnon Battle Cruisers

The Klarnon Battle Cruisers are less powerful and less sophisticated than the Endeavor, but usually appear in squadrons of several ships at a time. Therefore the overall Klarnon firepower in a battle will usually exceed the firepower of the Endeavor. The captain of the Endeavor must use skill and cunning to overcome the Klarnons' advantage.

At the beginning of battle each Klarnon ship is equiped with 8ØØ units of energy and 3 photon torpedoes. The Klarnon energy may be used for phaser fire, for movement, or for absorbing fire from the Endeavor. When a Klarnon ship's energy is reduced to zero, it has been destroyed.

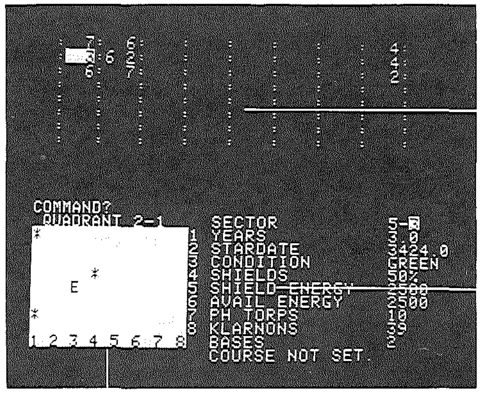
The Klarnons have both phasers and photon torpedoes. These weapons work much like those of the Endeavor, but Klarnons have the disadvantage of not being able to LOCK, LOAD, or UNLOAD photon torpedoes.

Klarnon ships can move one sector unit of distance at a time during battle. If the Endeavor retreats from a quadrant before all Klarnons in it are destroyed, the Klarnons will probably be fully restored if and when the Endeavor returns. Klarnons will never enter a quadrant that is occupied by a starbase, however. The base could easily thwart a Klarnon attack.

Control Console Display

At this point you should return your attention to your Apple. If this is your first mission, accept the setup conditions given, by typing N for NO. When asked, type the Self-Destruct Password you choose and then press RETURN . In a few seconds the Endeavor's Control Console display will appear on the screen. When the display is complete, the Apple will signal you with a beep.

Through the Endeavor's Control Console you can communicate with your entire ship and obtain information on the condition of both your ship and the galaxy. The following photograph is an example of a typical Console display at the beginning of a mission.



```
COMMAND?
 QUADRANT 2-1     SECTOR         5-8
                1 YEARS          3.0
                2 STARDATE       3424.0
                3 CONDITION      GREEN
                4 SHIELDS        50%
                5 SHIELD ENERGY  2500
                6 AVAIL ENERGY   2500
                7 PH TORPS       10
                8 KLARNONS       39
1 2 3 4 5 6 7 8   BASES          2
                  COURSE NOT SET.
```

At the beginning of the game, the top portion of the screen is occupied by the Galactic Record. The Galactic Record is an 8 x 8 array representing the 64 quadrants that make up the Galaxy. This record provides readings that tell you the number of Klarnon ships, Federation bases, and stars that occupy each quadrant that has been observed by the Endeavor so far. Observed quadrants include all quadrants the Endeavor has occupied and, if the long range sensor sub-system is operational, all 8 quadrants adjacent to these. The quadrant you are currently occupying is shown in inverse mode (black print on a white background).

The 3 possible readings are shown in the following manner:

```
              Number of bases
  Number of Klarnons  |  Number of stars
                   \  |  /
                    :3 5:
```

Because Klarnon ships never enter a quadrant that is occupied by a Federation base, at most 2 digits will be given for each quadrant. A blank space is equivalent to a reading of Ø . For example, there are no Federation bases in the quadrant shown in the example above.

Here are some more examples of quadrant readings:

```
:3 5:   3 Klarnons, no bases, 5 stars

: 13:   1 base, 3 stars

:  4:   4 stars
```

The Galactic Record will also be displayed whenever the NAVIGATION command is issued. As long as it is not destroyed in combat, the Galactic Record will be maintained.

Quadrant Display

The white square in the lower left corner of the screen is the Quadrant Display. This display is a blowup of the white square in the Galactic Record. It shows the locations of all spacecraft and other objects in the quadrant the Endeavor is currently occupying. Figure 4 below is an example of a typical Quadrant Display. North-South coordinates are on the right border and East-West coordinates are on the lower border. (Any time you are asked to input coordinates, remember that the first digit becomes the North-South coordinate and the second becomes the East-West coordinate.) The galactic coordinates of the quadrant are given at the top of the Quadrant Display.

In order to use the Quadrant Display, you must know the code for reading it. The code is very simple. The Endeavor is represented by E , a Klarnon ship by K , a star by * , and a base by B . During battle, a photon torpedo is represented by # and a Klarnon ship firing phasers is represented by K in normal video mode (white print on a black background).

```
        QUADRANT 5-1
 ___________________________
|                           | 1
|                           | 2
|                       *  E| 3
|          *                | 4
|              K            | 5
|  *   *                    | 6
|                           | 7
|                  *        | 8
| 1   2   3   4   5   6   7 | 8
|___________________________|
```

Figure 4
Quadrant Display

Status Display

The current status of the Endeavor is shown in the lower right portion of the screen. The Status Display on your screen probably looks something like the one in Figure 5 below.

```
SECTOR                5-6
YEARS                 3.Ø
STARDATE              3424.Ø
CONDITION             GREEN
SHIELDS               5Ø%
SHIELD ENERGY         25ØØ
AVAIL ENERGY          25ØØ
PH TORPS              1Ø
KLARNONS              31
BASES                 4
COURSE NOT SET.
```

Figure 5
Status Display

The first line of the display gives the coordinates for the SECTOR occupied by the Endeavor. The second line indicates the YEARS remaining in the Endeavor's mission, and the third line indicates the current STARDATE.

The Endeavor's CONDITION is GREEN if it has enough energy to ensure safety and if there are no Klarnons in the current quadrant. If energy is dangerously low, CONDITION YELLOW will flash, and if there are Klarnons in the quadrant (which is the same as saying that the Endeavor is under attack) CONDITION RED will flash. Despite appearances, CONDITION YELLOW can be more threatening than CONDITION RED.

SHIELDS indicates the percentage of the Endeavor's total energy (excluding photon torpedoes) allocated to its shields. SHIELD ENERGY indicates the energy the Endeavor can safely absorb from Klarnon attacks. If SHIELD ENERGY falls below 1ØØ units, the Endeavor may be damaged. If it falls below zero, the Endeavor will definitely suffer damage. AVAIL ENERGY shows the energy the Endeavor can use for phaser fire, propulsion, and operation of the ship's subsystems.

PH TORPS shows the number of unused photon torpedoes. At the beginning of your mission, the Endeavor will have 1Ø. KLARNONS shows the number of Klarnons still to be destroyed in this mission, and BASES shows the number of unused replenishment bases. Each base can be used only once during each mission.

The last line reads COURSE NOT SET . When you are firing under the control of the Endeavor's Apple 81 computer, you can set a strategic destination to move toward as you are firing. COURSE will then show the coordinates of the sector the Endeavor will move toward while firing.

Conversational Display

Your commands and their consequences are indicated directly above the Quadrant Display in the section of the screen set aside for the Conversational Display. Immediately above the Quadrant Display you should see COMMAND? . This means that the Endeavor is ready for your command. During battle and in some other instances, the Conversational Display gets too large for the space allotted for it. When this happens the Conversational Display will use the space now occupied by the Galactic Record.

Command Display

To see the Command Display, wait for the COMMAND? prompt and then press any letter key on the Apple keyboard. The list of available commands will be displayed in the lower right of the screen in place of the Status Display. The display appears as follows:

```
1  NAVIGATION
2  SHIELD ENERGY
3  DAMAGE RPT
4  PHASERS
5  PH TORPS
6  LOAD PH TORPS
7  COMPUTER
8  PROBE
9  SELF-DESTRUCT
```

Figure 6
Command Display

To select one of the 9 commands, enter the number to the left of the command. Your Apple will ask you for any additional information required to execute the command.

Following is a discussion of the commands available.

1 NAVIGATION

The NAVIGATION command is used to move the Endeavor to a different sector within a quadrant or to a different quadrant in the Galaxy. After pressing 1 in response to COMMAND? , the Galactic Record will be displayed in the upper portion of the screen.

If the warp drive is not damaged, the question
WARP OR ION DRIVE (W OR I)?
will appear. If you type W , then the question
WARP (1-7)?
will appear. Type a number from 1 to 7 to choose the WARP FACTOR.

If you type I in response to
WARP OR ION DRIVE (W OR I)?
the question
DURATION (1-9)?
will appear. Type a number from 1 to 7 to choose the DURATION.

If the WARP DRIVE is damaged, the question
WARP OR ION (W OR I)?
will be replaced by the message
ION DRIVE ONLY DURATION (1-9)?
If any letter key is typed, the NAVIGATION function is aborted and COMMAND? appears above the Quadrant Display again.

After the WARP or DURATION has been given then the question
COURSE (Ø-359)?
will appear. The direction chart is given below:

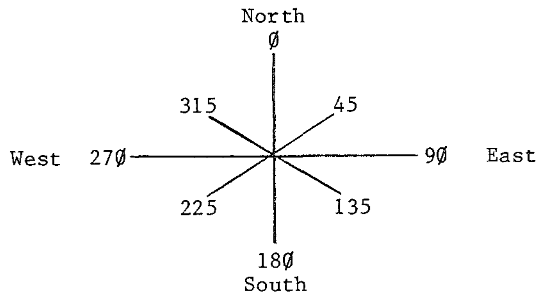


Figure 7
Direction Chart

To choose a course, type any number between Ø and 359, followed by a RETURN . If you type a letter key, the NAVIGATION command will be aborted and COMMAND? will reappear

2 SHIELD ENERGY

The SHIELDS parameter allocates the desired percentage of total energy to the shields, and the remainder to the Endeavor system functions. This value is set at 5Ø% at the beginning of your mission and may be changed with this command. Transfering energy takes time, however, so you have to plan ahead before you use the SHIELD ENERGY command.

3 DAMAGE REPORT
A display indicating the current status of all the Endeavor's systems is displayed at the top of the screen in place of the Galactic Record whenever the DAMAGE REPORT command is executed. If the subsystem is operational then the display will indicate OK . If the subsystem is damaged the display will indicate the estimated time to repair that unit. If dilithium crystals are destroyed, the display will indicate that too. A typical damage report is illustrated below.

```
WARP DRIVE              OK
SR SENSORS              OK
LR SENSORS              OK
PHASERS                 OK
PH TORPS    DAMAGED Ø.24 YEARS TO REPAIR
COMPUTER                OK
PROBE                   OK
```

Figure 8
Damage Report

(One year is the same as one stardate.)

4 PHASERS
The PHASERS command will initiate the phaser firing sequence. If phasers are not LOCKed on a target,
PHASERS READY: ENERGY TO FIRE?
will appear on the screen. Type the number of energy units you wish to fire and press RETURN . The phaser fire will be divided evenly between all Klarnon ships in the quadrant unless the number given is larger than the amount of energy the Endeavor has available. In that case the computer will politely decline to fire, as the Endeavor would be blown up in the process.. (If the computer is damaged....) When you type a valid number, the phasers will fire.

If phasers are LOCKed onto a target or targets, you will be asked
AUTO OR MANUAL (A OR M)?
Type A or M , and
PHASERS READY: ENERGY TO FIRE?
will appear on the screen. The LOCKed phasers will fire only at those targets that have been LOCKed into. Answer with the amount of energy you wish to expend, and the battle will commence.

If no Klarnons are in the quadrant, the phasers will not fire.

If a course has been set and you choose AUTOmatic fire, the Endeavor will move while firing. This feature is handy for avoiding enemy photon torpedoes. The Klarnons have no such automatic fire.

5 PH TORPS
The PH TORPS command will initiate the photon torpedo firing sequence. If the photon torpedoes are not LOCKed, you will see
PH TORPS NOT LOCKED
TRAJECTORY?
After you reply, with a number from Ø to 359, one photon torpedo will be fired in that direction.

If photon torpedoes are LOCKed, you have the option of automatic fire. The display will look like

```
PH TORPS    LOCKED ON
1-6 2-7 3-4 4-6 6-6
AUTO OR MANUAL (A OR M)?
```

If you reply M , you must fire manually by giving the desired trajectory when asked. If you reply A , all preset torpedoes will be fired at their targets (unless there are more targets present than photon torpedoes, in which case the computer will complain). If a course has been set, the Endeavor will move while firing.

6 LOAD PH TORPS
This command causes available energy to be transformed into photon torpedoes, or vice versa. This conversion uses 1ØØ energy units for each photon torpedo loaded or unloaded. It is possible by blow up the Endeavor by trying to LOAD more photon torpedoes than you have energy for, so use caution when using this command. To LOAD, type +1 (to LOAD 1 torpedo) to +6 (to LOAD 6 torpedoes). To UNLOAD, type -1 to -6 . To abort the command, type Ø .

7 COMPUTER
Command 7 sets the Apple 81 on-board computer in operation. See the section on Apple 81 computer instructions for details.

8 PROBE
The PROBE command sends a probe out into the quadrant currently occupied by the Endeavor. This probe reports the status of all Klarnon ships in the quadrant: energy levels, photon torpedoes left, and whether targeted by the Endeavor's automatic weapon systems. The probe's information is displayed at the top of the screen where the Galactic Record was when you began. A typical probe display is illustrated below.

```
COORD     ENERGY     PH TORPS     LOCK

 3-1       54Ø           2       PHASERS
 6-3       21Ø           3
 8-5       62Ø           2       PH TORPS
```

Figure 9
Probe Display

9 SELF-DESTRUCT
Command 9 starts the self-destruct sequence: if it is followed by the correct password, the Endeavor will be destroyed with all hands lost. If sufficient energy is available, all enemy vessels in the quadrant will also be destroyed.

Apple 81 On-Board Computer Instructions

The Apple 81 computer enables the Endeavor to make navigational calculations, and controls the ship's weapon systems for maximum effectiveness in combat. It enables the ship to take evasive action while firing, thus making it a match for a squadron of Klarnon vessels. When the computer has been invoked by typing 7 in response to COMMAND? , you will see

```
APPLE-81 HERE
WHAT IS YOUR INSTRUCTION?
```

To execute one of the computer instructions, type a number from 1 to 6. Typing a letter key will display the list of instructions. The computer instructions are described below.

1 *COMPUTE QUADRANT
This instruction plots the shortest path to any desired quadrant. When you see COORDINATE ?- , type the coordinates of the destination, and the COURSE and WARP needed to reach that location will be displayed.

2 *LOCK PHASERS
This instruction LOCKS the phasers onto the Klarnons you choose to attack. The Klarnon ships will be tracked by the phasers if the enemy takes evasive action. You will be asked for the # OF TARGETS, and then for the COORDINATEs of each target. Incorrect data will give you a <DATA ERROR> message. The phasers will be locked onto all targets whose coordinates are entered. If no valid coordinates are entered, the instruction will be aborted.

3 *LOCK PH TORPS
This instruction causes the photon torpedoes to LOCK onto the Klarnons you have chosen. These ships will be tracked if they take evasive action. This instruction works just like the *LOCK PHASERS instruction.

4 *LOCK COURSE
Instruction number 4 LOCKs the COURSE that will be followed during automatic fire in combat. You will be asked to enter each COORDINATE of the chosen destination. The COURSE will be displayed at the bottom of the status display. During each round of combat, the Endeavor will move one sector unit toward the destination.

5 *COMPUTE TRAJECTORY
When you give this instruction, the Apple 81 will ask you for the COORDINATEs for a particular target and then give you its TRAJECTORY and RANGE. If you type the COORDINATEs of a destination, the TRAJECTORY and RANGE displayed will be the COURSE and DURATION for Ion Drive motion.

6 *RETURN
This instruction puts you back in the command module.

COMMAND SUMMARY

1.	NAVIGATION	Moves starship with warp or ion drive Warp (W) drive moves 1-6 quadrants WARP FACTOR?- (1-6 quadrants) COURSE?- (1-359 degrees) Ion (I) drive moves 1-9 sectors COURSE?- (1-359 degrees) DURATION?- (1-9 sectors)
2.	SET SHIELD ENERGY	Sets percentage (Ø-1ØØ) of energy allocated to deflector shields
3.	DAMAGE REPORT	Displays status of subsystems
4.	PHASERS	Starts phaser firing sequence AUTOMATIC OR MANUAL?- (A or M) ENERGY TO FIRE?- (Ø-Avail energy)
5.	PHOTON TORPEDOS	Starts photon torpedo firing sequence AUTOMATIC OR MANUAL?- (A or M) TRAJECTORY?- (Ø-359, manual only)
6.	LOAD PHOTON TORPEDOS	Loads or unloads photon torpedoes Load: (+1 to +6) Unload: (-1 to -6)
7.	COMPUTER	Turns on Apple 81 computer
8.	PROBE	Sends out probe of current sector
9.	SELF-DESTRUCT	Starts self-destruct sequence ENTER SELF-DESTRUCT PASSWORD-

APPLE 81 ON-BOARD COMPUTER SUMMARY

1.	COMPUTE QUADRANT	Calculates course and warp factor to desired quadrant
2.	LOCK PHASERS	Set phasers to track targets # OF TARGETS?- (1-Klarnons) COORDINATE ?- 1-1 to 8-8
3.	LOCK PHOTON TORPEDOS	Set photon torpedoes to track targets # OF TARGETS?- (1-Klarnons) COORDINATE ?-
4.	LOCK COURSE	Set evasion course for automatic fire
5.	COMPUTE TRAJECTORY	Calculates trajectory and range (course and duration) to desired sector
6.	RETURN	Return to command module